Methodology for Evaluation of Collaboration Systems

by
John Cugini
Laurie Damianos
Lynette Hirschman
Robyn Kozierok
Jeff Kurtz
Sharon Laskowski
Jean Scholtz

The Evaluation Working Group of
The DARPA Intelligent Collaboration and Visualization Program
Revision 3.0: August 27, 1997 Draft
Submit or view comments at
http://www.antd.nist.gov/~icv-ewg/forum.html

Contents

List of Tables

List of Figures

Preface

In addition to the listed authors, who contributed directly to this draft, thanks are in order to other members of the EWG for their contributions to the EWG meetings and discussions. Special thanks to Doug Montgomery and Stephen Nightingale for the initial draft and especially to Charles Sheppard for formatting and maintaining the document on line.

1 Introduction

The DARPA Intelligent Collaboration and Visualization program (IC&V) has the goal of developing the generation after next collaboration middleware and tools to enable military components and joint staff groups to enhance the effectiveness of collaborating problem solvers through:

- Gathering appropriate problem solvers together across time and space for rapid response in time critical situations; and
- Bringing appropriate information resources together across time and space within the context of a task.

The IC&V program has funded a number of groups to develop collaborative technologies to address these problems; it has also devoted a portion of the funds towards the establishment of evaluation metrics, methodologies and tools. Since the technologies developed are likely to be diverse, it seems appropriate to review the original program objectives and use them as a basis for establishing the direction which the investigation of evaluation methods might take. The IC&V program objectives are:

1. Enable access via diverse portals, from hand-held through room-sized;
2. Enable interoperability across diverse encoding formats, coordination and consistency protocols, and real-time services;
3. Scale collaborations to 10 active contributors, 100 questioners, and 1000 observers;
4. Reduce by an order of magnitude the time needed to generate collaborative applications;
5. Enable real-time discovery of relevant collaborators and information within task context;
6. Reduce by an order of magnitude the time to establish collaborative sessions across heterogeneous environment;
7. Reduce by an order of magnitude the time to review collaborative sessions; and

8. Improve task-based performance of collaborators by two orders of magnitude.

The effectiveness of the overall IC&V program will be evaluated with respect to these high-level objectives. The Evaluation Working Group (EWG) of the IC&V program has been established to support implementation of this evaluation. The role of the Evaluation Working Group is to develop the metrics and evaluation methodology, and to develop, or guide the development of, specific tests and tools for achieving effective and economical evaluation for the collaborative technologies that make up the IC&V program.

1.1 The Evaluation Working Group and its Aims

The Evaluation Group comprises researchers from several sites (CMU, MITRE, NIMA, NIST and Amerind), with diverse backgrounds and interests.[1] The EWG has taken as its primary task the definition and validation of low-cost methods of evaluating collaborative environments, such that any researcher in the collaborative computing research community can use these methods to evaluate their own or other research products. This objective is further refined into a set of goals as follows:

1. To develop, evaluate and validate metrics and methodology for evaluating collaborative tools.
2. To provide reusable evaluation technology, such that research groups can assess their own progress.
3. To provide evaluation methods that are cheap relative to the requirements.

[1] The IC&V EWG currently has members from Carnegie Mellon University (CMU), James Morris and Christine Neuwirth, PIs, with a focus on the effect of computer tools on group problem solving and communication, especially collaborative writing; the MITRE Corporation, Lynette Hirschman, PI, with a focus on logging and data collection for evaluation and training; the National Imagery and Mapping Agency (NIMA), PI Kim Walls, with a focus on technical and usability evaluation of CSCW technology; the National Institute of Science and Technology (NIST), Sharon Laskowski, PI, with a focus on the testing of methodologies and tools for visualization, networking, and distributed systems; and Amerind, Lloyd Brodsky, PI, with a focus on system dynamics modeling for evaluation of collaborative planning systems. Several other groups have expressed interest in attending meetings. Lynette Hirschman, MITRE, chairs the EWG.

4. To apply DOD-relevant criteria, e.g., evaluate using scenarios drawn from, e.g.,
    - Planning/design/analysis domains.
    - C2 environment to capture planning/reaction/replanning cycle.
    - Disaster relief exercises.
    - Collaborative information analysis activities.
5. To define an application vision that will drive collaborative computing research.

The technologies supported under the IC&V program range from infrastructure technologies at the level of networking and bus protocols, to middleware to provide easy interoperability, to user-oriented collaborative tools. Given this wide range of technologies and the background of the EWG members, the EWG has decided to focus on the user-oriented end of the spectrum. In addition, specific interests of various EWG members (NIST, in particular) may lead to subgroups working in the area of infrastructure technology evaluation, especially as these areas affect the user level (e.g., sensitivity to network load may limit number of participants in a collaborative session). Currently, there are no plans for the EWG to provide evaluation metrics aimed at the software infrastructure, e.g., how easy it is to make a new application collaborative, or how a given layer of middleware might enhance interoperability. These are clearly important issues that will affect the long-term success of the program, but they lie outside the scope of the EWG as it is currently constituted.

1.2 The Scope and Structure of this Document

This methodology document has been developed as the vehicle for encoding agreements of the IC&V Evaluation Working Group as we develop a methodology for evaluation of the IC&V technologies.

The set of problems which can be solved through collaboration is independent of any of the efforts funded under the IC&V program, but it is presumed that each technology in the program can be used to help solve some subset of the range of problems. The challenge for the EWG is to provide an evaluation methodology that can be applied across the diverse

IC&V research projects and approaches to collaboration. It is critical to provide researchers with tools to measure their own incremental progress, as well as providing methods to evaluate the impact of specific technologies on the overall effectiveness of collaboration.

We outline here a particular approach to evaluation, namely *scenario-based evaluation*. The goal is to develop a suite of scenarios that are scripted for a problem-solving community and enacted using the technologies under evaluation. Since the technologies are diverse, the scenarios must be carefully crafted to be generic enough that they are capable of providing meaningful evaluation across multiple research projects. Enaction of the scenarios is used to provide data for the functional evaluation, or to exercise tools developed for the technology evaluation. Different scenarios will exercise different aspects of collaborative work ranging from number of participants, to kind of shared objects, to how participants need to interact with each other and with the shared objects.

The remaining sections of this document are structured as follows. Section 2 situates this methodology in the context of current evaluation approaches from human-computer interface (HCI), computer-supported collaborative work (CSCW) and spoken language systems (SLS) research, and discusses the rationale for scenario-based evaluation. It also defines critical terminology for use in the remainder of the document.

In section 3, we present a framework that defines the design and implementation space for collaborative systems. We define a set of generic task types that can be used to construct scenarios.

Section 4 discusses the concept of a *scenario* as a vehicle for simulating a collaborative activity. The exercise and evaluation of any specific collaborative technology requires selection of appropriate scenarios. The section describes a scenario repository and methods for using scenarios, such as iterative evaluation, assessment of system appropriateness, and comparison of systems.

Section 5 discusses a range of metrics and measures for evaluating collaborative technologies at various levels and illustrates these with several examples.

Finally, section 6 concludes with some suggestions for future directions, including the possibility of (partially) automating participants in scenarios, to limit subject variability and reduce the human effort required to

collect data. The reader can view a template for generating scenarios at `http://zing.ncsl.nist.gov/~cugini/icv/domain-template` and some sample scenarios at `http://www.antd.nist.gov/~icv-ewg/pages/scenarios.html`.

2 Background

This section provides background for the remainder of this document. The first subsection looks at related evaluation efforts from the HCI, CSCW and SLS communities. The second subsection defines our terminology, and the third subsection lays out the scenario-based approach in greater detail.

2.1 Related Evaluation Approaches

Evaluations of human-computer interaction have traditionally been done by a number of methods, including field studies, laboratory experiments, and inspections. Each method assesses different aspects of the interfaces and places different demands on the developer, user, and evaluator.

2.1.1 Methods of Interface Evaluation

Evaluations of collaborative technology are done best through field evaluations because they can, among other things, be used to assess social-psychological and anthropological effects of the technology (Grudin, 1988). Field studies unfortunately require a robust system, an environment that can accommodate experimental technology, and substantial funding. These three conditions are difficult to satisfy, and so researchers turn to less onerous means, such as inspection methods and laboratory exercises.

System inspections, such as cognitive walk-through, heuristic evaluation, and expert inspection, employ a set of usability guidelines written by usability experts. There exists a fairly large set of guidelines for user interface design and single user applications, but few guidelines are available for multi-user applications or collaborative technology. Also, these methods are largely qualitative, not quantitative, and require HCI expertise which may not always be available.

This leaves laboratory experiments, or empirical testing, which as an evaluation technique more closely relates to field studies than inspection techniques. Experiments are very appealing for new and rapidly evolving technology and are less expensive than field studies. However, since they are conducted in controlled environments with time restrictions they less accu-

rately identify dynamic issues such as embedding into existing environments, learning curves, and acculturation. [2]

The process that we present here cannot presuppose a specific work context. Rather, we have chosen to develop scenarios and measures based on high-level collaborative system capabilities, to provide broad applicability across a range of applications. Ethnographic studies related to specific work contexts could provide a useful addition tool for validating the measures, because some measures may not be appropriate or of interest in certain contexts.

2.1.2 Use of Scenarios

Scenarios are used in laboratory experiments to direct system usage. They are a versatile tool that can be used for many development activities including design, evaluation, demonstration, and testing. When used for evaluation, scenarios exercise a set of system features or capabilities.

We would like to define general, reusable scenarios for collaborative technologies. This is a challenge, requiring consideration of a large set of technologies, but we can build on earlier work in this area.

In 1995, researchers met at EHCI to develop scenarios that could be used to design, evaluate, illustrate, and compare CSCW systems (Bass, 1996). Their approach was to define generic tasks that would be used to construct technology specific scenarios. The tasks they described were mostly single user activities like joining a conference, setting up a new group, and integrating a system with a collaborative tool.

Our approach (see below) defines *collaborative system capabilities* or functional requirements that are used to decompose scenarios into a task/capability matrix. Our "capabilities" are defined at a higher level than the tasks defined by the EHCI researchers. The tasks we use to create the matrix are taken from the scenarios that we define which are much more general than those described by the EHCI group. Consequently, they are appropriate vehicles for cross-technology comparisons. Many of them can also be decomposed into subtasks if the whole scenario is too large to use for this

[2] Watts et al. (1996) recommend compensating for this flaw by performing ethnographic studies. They describe a process of evaluation that begins with ethnographic studies followed by laboratory experiments. The ethnographic study helps evaluators understand the "work context" which influences what is measured and observed in the laboratory.

purpose. Also, general scenarios can be instantiated for any system that supports the capabilities required by the scenario with little change to the scenario.

Nardi (1995), who has extensively studied the use of scenarios for interface design, has argued for a provision to create a library of reusable scenarios. We will begin to populate such a library with our capability-level scenarios. These scenarios are technology independent. Technology specific scenarios will be added when the scenarios are specialized for real systems.

In order for a scenarios to be reusable we must design them with "salience." Potts (1995) emphasizes the importance of identifying goals and obstacles in developing salient scenarios. We are using these two attributes to characterize our scenarios. Each scenario will have a set of goals and obstacles demonstrated or exercised in a scenario. The evaluator using our scenario library will begin by identifying the goals and obstacles addressed by the system under evaluation. From that characterization the evaluator can select the appropriate scenario.

The sections that follow will clarify our methodology for defining scenarios for reusability and selecting scenarios for evaluation.

2.2 Definition of Terms

We find it necessary to pause at this point to define the terminology we will use in the remainder of this report.

This evaluation program is concerned with the three principle variables of people, collaborative environments and collaborative activities. We know that people are the human actors who engage in collaborative activities. But how can we classify the collaborative environments and activities in meaningful ways? What levels of abstraction are useful to our discussion of evaluation?

2.2.1 Collaborative Environments

A *collaborative environment* is a system which supports users in performing tasks collaboratively. It may be a particular piece of software, such as

Lotus Notes, or MITRE's Collaborative Virtual Workspace, or it may be a collection of simpler components used together to enable collaboration.

We are charged with the task of providing an evalutation methodology for collaborative environments, present and especially future. Part of our approach involves examining the types of things an environment allows one to do in support of collaboration. To describe them, we must define the terms *capability, service,* and *technology.*

Collaborative *capabilities* are relatively high-level requirements imposed upon a collaborative environment in order to support users in performing particular collaborative tasks. Examples are things like synchronous human communication, persistent shared object manipulation, archival of collaborative activity, etc.

The term *service* is used to describe the means by which one or more collaborative environments achieve a given capability, and *technology* is used to describe the the particular hardware and/or software implementation of a service.

So for example, for the collaborative *capability* of *synchronous human communication*, one *service* which may is used to achieve it in a variety of collaborative environments is *audio conferencing.* One *technology* for doing audio conferencing is LBL's *Visual Audio Tool.*

Examining which capabilities a collaborative environment supports, and the services and specific technologies it uses to do so is one way of generating a functional categorization of the collaborative environment, which can be used to determine more easily which types of activity it is best suited for.

2.2.2 Tasks or Collaborative Activities

Tasks or *collaborative activities* are the things which people do, or wish to do, collaboratively.

We use the term *task* synonymously with *collaborative activity.* This is a term that transcends the level of detail at which the activity is described: *task* may refer to anything from a real activity that people are engaging in to a highly scripted mock-up of such an activity intended strictly for evaluation purposes. We also talk about *subtasks* which are smaller chunks that a task

may be broken down into.

We may utilize several different types of description of mocked-up activity in the course of our evaluations. At the highest level we have the *scenario*. This is a high-level description of an artificial collaborative task that subjects might be asked to engage in to test out collaborative systems. A *fully specified scenario* will include the background documentation, instructions, etc. required to actually have a subject or subjects enact the scenario. Scenarios are often broken down into their constituent tasks. In some cases it will be possible to do evaluations based upon only a subset of the tasks for a given scenario.

2.2.3 Scripts

Some evaluation goals might be better met by scenarios which are precisely repeatable, perhaps even by automated actors. For this purpose we have *scripts* which are procedural descriptions of scenarios. Scripts can be written at levels corresponding to technologies, services, or capabilities.

A *technology-level script* specifies all the details of the enactment of the scenario, including how each step should be completed on a particular technology. This type of script would allow users unfamiliar with the collaborative environment being evaluated, or even automated participants (*participatons*) to play roles in the script.

A *service-level script* specifies the steps in the scenario in terms of the services to be used, without committing itself to a particular technological implementation of the service. It may, for example, state that a particular discussion is to take place via videoconferencing, without specifying exactly how that will be achieved in any particular system.

A *capability-level script* gives the detailed instructions in terms of the capabilities that the script uses. This script does not require the use of any particular services, so it can be run on any system that provides some means of instantiating the required capabilities. This last level makes it possible to define scripts which can even be used across platforms which provide radically different sets of services in support of the same basic capabilities.

Section 3 presents a framework of collaborative tasks, capabilities and services in greater detail.

3 A Framework for Collaborative Systems

3.1 Introduction

Our goal is to describe a given collaborative system and to evaluate how well that system supports various kinds of collaborative work. The framework outlined in this chapter describes the design and implementation space for collaborative systems used in a work environment.

Groups should be able to use our framework to describe their requirements and to ascertain how well a given CSCW system supports their work. Developers of CSCW systems should be able to use our framework to describe their system and to determine the types of group work that could be easily accomplished using the system.

This framework builds on a framework devised by Pinsonneault and Kraemer (1989) to analyze the impact of technology on group process while controlling for the effect of other contextual variables.We have merged the work of McGrath(1984) into our expanded framework to enable us to classify task that groups perform.

In this section, we give an overview of the framework and we describe how the framework can be used to describe and evaluate CSCW systems. We also provide detailed descriptions of each level of the framework.

3.2 Overview of the Collaborative Framework

The framework is divided into four levels: requirement, capability, service, and technology.

The *requirement level* of the collaborative framework describes the requirements of the group with respect to the tasks being performed by the group and the support necessitated by the characteristics of the group. The tasks described in the framework include work tasks as well as transition tasks. Requirements for supporting different types of groups include support for the social interactions of the group as well as the requirements due to group size, location, computer platforms, etc. The requirement level is divided into four sections: work tasks, transition tasks, social protocol requirements, and group characteristics.

The *capability level* of the framework describes functionality that is needed to support the different requirements. The capabilities can be divided into subsections that correspond to the four subsections of the requirement level. The functionality described in capabilities can be obtained from different services. For example: the capability to have a side chat with another meeting participant during an electronic meeting could be accomplished by a text chat service or by telephone service. Certain capabilities may be necessary or recommended to support work and transition tasks, social protocols, and group characteristics in the requirement level.

The *service level* describes services such as e-mail, audio, video, application sharing, networking services, etc. that can be used to support the capabilities needed in CSCW systems. Different types of services can be used to provide the same capabilities needed to support specific requirements. Comparisons and tradeoffs of performance and cost can be made at this level.

The *technology level* describes specific implementations of services. This level could be considered as the set of all possible components needed to build a given CSCW system, including integration and user interface components. For example, all different e-mail systems would exist at this level, as would the numerous ways to implement floor control, and the various algorithms to control documentation locking and requesting, and the various networking services such as ATM. Specific implementations can be compared with respect to performance, cost, functionality, and usability.

3.2.1 Using the Framework to Evaluate a CSCW System

The framework can be used for evaluation either in a top-down (requirement level to technology level) or bottom-up fashion. Table 1 gives an overview of the questions asked in each mode of evaluation.

The framework used in either mode of evaluation allows the evaluator to select a subset of the available systems or to determine if a single system will support the group adequately. Systems that do not meet or exceed acceptable levels for the measures available at a given level can be eliminated.

At the requirement level we can evaluate how well a CSCW system supports the work tasks, the transition tasks, the social protocols and characteristics of a specific group. We can also evaluate the artifacts produced as well as

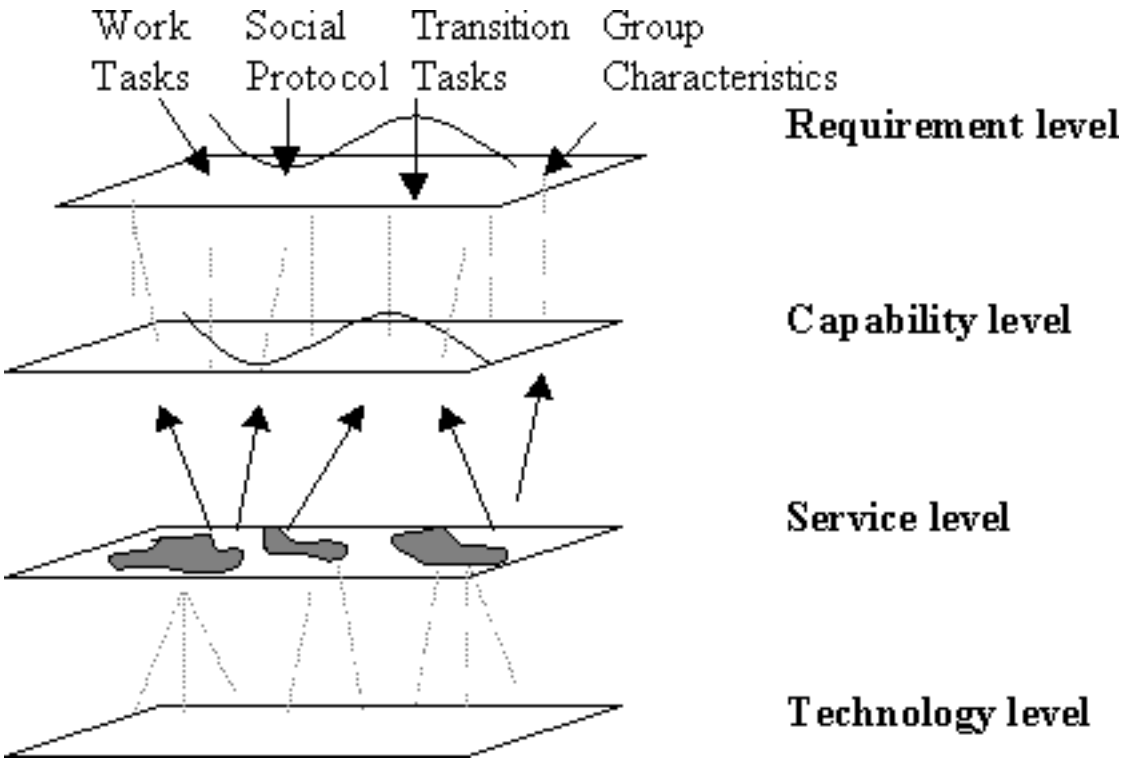

Figure 1: The collaborative framework

the process the group used in producing an artifact. The detailed framework description (Section 3.3) describes the measures for task outcome for each task type. It is important to note that the variables being measured differ depending on the type of task the group is doing, that is, different measures are needed to evaluate the outcome of a brainstorming task than to evaluate the outcome of a planning task.

Group process refers to the way the group works. Examples of measures of group process include time to reach a decision, efficiency of communication and equality of participation. Depending on the work task being perfomed, some measures are more relevant than others. (See Section 5 for details). For example, equal participation by group members is important in brainstorming tasks. Therefore, groups that work well in brainstorming should have a high rating for equality of participation. Equal participation is not as necessary in a lecture situation. Therefore, ratings on equality of participation from individuals involved in lecture types of collaboration would not necessarily be rated high even though the lecture was very successful.

At the capability level, we are interested in evaluating how well given capabilities support the work tasks, transition tasks, social protocols, and characteristics of the group. To obtain answers to these questions, measures such as time on task, awareness questions, amount of setup time for equipment and configuration, are used.

At the service level, we can use the functionality of various types of services to understand how a given capability would be supported using a particular type of service. This allows us to compare sevices to determine which service seems most appropriate given the characteristics of a group. Performance thresholds, including robustness, can be examined at this level. Tests, such as the ability to view an image with a certain amount of discrimination and the acceptability of audio or video, could be conducted at this level.

At the technology level, specific implementations exist. Therefore, actual usability measures can be examined for an implementation. The implementations of the services can be evaluated to make sure they meet threshold values. Performance comparisons can be made at this level between different implementations.

3.2.2 Constructing Scenarios at the Requirement Level

The previous section discussed the types of questions that should be answered in order to evaluate CSCW systems. In this section, we address the process of answering those questions. The best way to evaluate a system is to use it. In order to compare several sytems, we need to make sure that they are used in a similar fashion. This framework has been designed to support scenario based evaluations, thus allowing evaluations in a real-work context that are repeatable in order to make comparisons across systems. Section 4 gives a detailed description of using scenarios for evaluation.

Scenarios are constructed by putting together tasks based on the work task types described in the requirement level of the framework. A group selects the various task types they will be using the CSCW system for, the social protocol parameters appropriate for the group and the type of transition tasks needed. The work task types are generic and thus, to construct scenarios, each group will need to instantiate a generic task type with objects from the work domain of the group. Transition tasks will be more fully described in the requirement level detailed description (Section 3.3.1), but they can be viewed as overhead tasks: cleaning up from one meeting activity and moving on to the next, assigning responsibilities to group members, arranging meeting times, notifying members, etc.

For example, here is a scenario that could be representative of what a group might do:

> "The human resource group meets at the usual group meeting time, decides upon the location of the Christmas party from the list suggested previously. They then decide upon the menu from the possibilities provided by the caterer and appoint Susan to call the caterer. Then they brainstorm on ideas for entertainment. The meeting ends after one hour. They decide to meet at the same time next week."

In the following descriptions T $_{i}$ refers to a type of transition task i. Task Type # refers to a work task type of that number. P $_{j}$ refers to a collection j of attributes for a social protocol. Using the collaborative framework to describe this session, we would say:

The group starts with a transition task to begin (T_{start}). They perform a decision-making task with no known solution (Task Type 4) using an appropriate social protocol (P_{chair}). They summarize the decision in the transition task ($T_{summary}$) and move to the next task which is another instance of a decision making task with no known solution (Task Type 4) and use the same social protocol (P_{chair}). The group transitions using a transition task to assign action items ($T_{action\ item}$) to the brainstorming task (Task Type 2). A different social protocol ($P_{no\ chair}$) is used for this task. An ending transition task which summarizes the plans and selects the time for the next meeting is then performed ($T_{time\ plans}$).

More succinctly, we could write:

$T_{start} \rightarrow$ (Task Type 4, P_{chair}) $\rightarrow T_{summary} \rightarrow$(Task Type 4, P_{chair}) $\rightarrow T_{action\ item} \rightarrow$ (Task Type 2, $P_{no\ chair}$) $\rightarrow T_{time\ plans}$

3.3 Detailed Framework Description

In this section, we describe each level of the framework in detail.

3.3.1 Requirement Level

We describe the requirement level based on the four subsections: work task types, transition tasks, social protocols, and group characteristics.

3.3.1.1 Work Task Types

The task descriptions are based mainly upon tasks from group research (McGrath, 1984). We have added an additional task type that is necessary to form a comprehensive collection of current work practices. We have noted in the description of task type nine that it is not from McGrath's original set.

The tasks described by McGrath should be thought of as a continuum, not discrete tasks. Along this continuum there are four higher level types: gen-

erate tasks, choose tasks, negotiate tasks, and execute tasks. Within these four task types, generic tasks are numbered [1-8, original McGrath] with successive tasks related to each other. The task descriptions and examples should enable users of the framework to select generic tasks that are comparable to the type of work they do.

In our detailed description, each task type has suggested measures of task outcome. In addition, for many task types, research has uncovered problems that may occur with groups performing this type of task. In some instances, computer-mediated processes or other group processes may be able to alleviate these problems. We have listed known problems so that comparison measures can be made to see if there is any effect. Where there is research about computer-mediated work or group interactions and the effect on the task outcome, we have included it under the heading "known research." An example of each task is also provided to help in understanding how the generic task maps to a real world task. Under "suggested capabilities" are listed those capabilities which the research suggests may be valuable in carrying out that particular generic task.

Type 1: Planning tasks (McGrath)

Group members are given a goal (previously chosen objective) and asked to develop a written plan for carrying out the steps needed to reach that goal. The written plan should include alternative paths or actions.

Specific measures for this type of task:

- Amount of time per participant
- Amount of calendar time
- Practicality of plan (quality of task outcome)

Known research:

- Social relations hinder task efforts
- There can be a strong effect on the group due to social influence and conformity.
- Groups often have trouble seeing alternatives - tend to focus on only a few alternatives.

- Participation can be very unequal. This increases as group size grows.
- Groups tend to avoid conflict and spend more time on non-controversial issues. Controversial issues tend to become personalized.

Example: The group has to produce an advertising campaign for a new account by the end of the month. They have a meeting to plan the different tasks that each member will carry out, complete with time lines for doing so and different coordination points.

Suggested capabilities:

- Calendar support
- Text object creation, editing, displaying, arranging, storing

Type 2: Brainstorming and group creativity

Members of a group are given a particular topic area and asked to brainstorm on ideas.

Specific measures for this type of task:

- Number of ideas
- Originality of ideas

Known research:

- Creativity of individuals is stifled by social influence of group
- Individuals are able to take advantage of creativity-enhancing forces in group - social support, cross stimulation

Example: The group has a goal to raise $200,000 to build a new community center. They generate ideas for funding raising events, people to ask for contributions and possibilities for loans or selling "shares" to the community members to raise this money.

Suggested capabilities:

- Anonymous communication
- Synchronous communication
- N way communication
- Shared workspace

Type 3: Intellective tasks

The group is asked to solve a problem for which there is a recognized solution. The group is asked to determine a concept, given instances of the concept. Alternatively, groups can be asked to generate an instance of a concept and are given feedback as to whether it is or is not the concept.

Specific measures for this type of task:

- Number of trials to solution
- Solution (quality of task outcome)
- Errors
- Inferred strategies

Known research:

- Written media is slower to arrive at a solution than voice media. But voice media uses more messages than written.
- Audio only does not differ significantly from face to face (and hence, probably video)
- Interacting groups are almost always more accurate than their average member
- Groups seldom do as well as their best members

Example: A logical reasoning problem such as the cannibal and missionary problem is an example of an intellective task with a demonstrable right answer.

The missionary and cannibal problem: Three cannibals and three missionaries are on a river bank. They need to cross to the other side. There is a row boat that can hold only two people. Only missionaries know how to row. At no time can there be more cannibals than mssionaries on either side of the river. What is the minimum number of trips that can be made to transport all six to the other side of the river?

Suggested capabilities:

- Shared workspace
- Gesturing, pointing, agreeing, disagreeing
- N way communication
- Private group communications

Variant: The twenty-question task

A target object is selected. A group or individual is asked to determine that object by asking a series of questions that can be answered by "yes" or "no". The goal is to determine the target object with as few "no's" as possible with a maximum of 20 allowed.

Specific measures for this type of task:

- number of problems correctly answered
- number of questions needed to answer each problem
- time to solution
- time-per-person to solution.

Type 4: Decision making task

Group members are asked to develop consensus on issues that do not have correct answers.

Specific measures for this type of task:

How far and in what direction (if any), the group as a group and the individuals in the group shift. [Attitudes are measured before and after group discussion]

Example: The group is asked to decide which of three possible job candidates should be hired. All candidates are computer science graduates with specialties in computer engineering. However, their work experiences are quite different. The group must decide which candidate would be the best match for the job.

Suggested capabilities:

- Shared workspace
- N way communication
- Side chats

Type 5: Cognitive conflict tasks

Members of the group hold different viewpoints. The group is asked to make a series of decisions from available information that is imperfectly correlated with criterion. The group has to arrive at a decision.

Specific measures for this type of task:

- Agreement among members
- Interpersonal trust
- Changes in member's views

Known research:

- Verbal interactions can lead to clarification of why group members are consistently using different policies. But if policies are used inconsistently, this leads to a distrust of and a reduction in understanding of the other.
- Group members may change policy to increase accuracy.

Example:

The group is hiring a designer for its User Interface Group. Three candidates are at the top of the list. One candidate is a computer scientist major with

some interface design experience, another is a psychology and human factors major, and the third is a graphics art and industrial design major. The group is divided about the type of experience best suited to this position. The group is interdisciplinary and each discipline tends to favor hiring the candidate most closely aligned with their discipline.

Suggested capabilities:

- Shared workspace
- N way communication

Type 6: Mixed motive tasks

Mixed motive tasks present a range of tasks, differentiated by the degree to which a group member's outcome is affected by a combination of his own actions and the group's outcome. McGrath also includes dilemma tasks in this category. Dilemma tasks involves a conflict of interest where the choices made by individuals or groups are combined to jointly determine the outcome. However, the collaboration between the individuals or groups is not directly related to determining the choices of each. Therefore, we have not included dilemma tasks in our framework.

Type 6A: Negotiation task: The group is divided into x subgroups with a negotiator elected for subgroup. The different subgroups disagree on an issue but an outcome has to be reached. Tradeoffs have to be made in multiple dimensions. It is not necessarily a zero-sum problem.

Specific measures for this type of task:

- Quality of solution as evaluated by each subgroup
- Time to solution
- Attainment of solution (task completion)
- Evaluations of negotiators by group
- Interpersonal relations between group members

Example:

Company A and Company B are negotiating the sale of supplies from A to B. Company A wants to sell more of the supplies at a lower price to company B, but this means that company B, while saving money on the sale, will have to arrange financing with company A.

Suggested capabilities:

- N way communication
- Group private communication
- Shared workspace
- Private workspace

Type 6B: Bargaining task: (suitable for two individuals)

A conflict of interest must be resolved among two individuals, but in this case whatever one individual gains results in a loss for the other individual. A tradeoff is made on a single dimension, such that what one party gains, the other party loses.

Specific measures for this type of task:

- Frequency of no-solution trials
- Absolute and relative gains of the two individuals over series of trials
- Responses to different strategies
- Opponents ratings of each other's bargaining strategies

Known research:

Negotiators are more competitive when any of these conditions hold:

- They think constituents distrust them
- They were elected
- They are being observed

- They have a prior commitment
- Their constituents belong to a highly cohesive group

Negotiators who do not belong to the group feel freer in the negotiation process but are less supported by the group.

Example:

There has been a price set by Company A for a machine that company B wishes to purchase. Company B does not feel that the price is low enough. Company A is trying to maximize profit as the company is having cash flow problems, but loosing the sale will be a problem also.

Suggested capabilities:

- N way communication
- Group private communication
- Shared workspace
- Private workspace
- Text object manipulation

Type 6C: Winning coalition tasks

Subsets of members make agreements. The subset that wins then allocates the resources among the group members. The two research questions are the formation of the coalition, and the allocation of the resources.

Specific measures for this type of task:

- Which coalitions form
- The division of outcomes
- Any shift over time

Example:

A variant of pachisi is played in which a player's piece is given a weight. Moves are based on product of the weight of the piece and the roll of a pair of dice. Players can play alone or can combine with one other player by adding their moves together. They must then agree on how to divide the payoff assuming they win.

Suggested capabilities:

- N way communication
- Side chats
- Shared workspace
- Private workspace
- Gesturing, pointing, agreeing, disagreeing
- Support for computational object
- 2D object manipulation

Type 7: Competitive performances

Groups are competing against each other with no resolution of conflict expected. Rather the goal of each group is to win over the other group. Subgroups are paired against each other an equal number of times, under an equal pattern of situations.

In the original McGrath work, these performances are physical. In this framework, these types of tasks may be physical or nonphysical performances.

Specific measures for this type of task:

- Team performance (quality of task outcome)
- Individual performance (quality of task outcome)
- The overall winner

Known research:

- Inter-group competition increases within-group cohesion.
- Success in a competitive task also increases within-group cohesion.
- Groups do not always distinguish between good group performance and winning.

Example:

A focal group competes with an opposing team that has a series of pre-planned, semi-standardized patterns. The responses to the focal group's moves are based on pre-planned strategies. A reconnaissance patrol or defense of a position is an example of such an activity.

Suggested capabilities:

- N way communication
- Side chats
- Private communication
- Secure communication
- Private (to group) workspace

Type 8: Non -competitive contests - against standards

Groups perform some sort of complex group task. The plan for the task has already been decided upon. In this type of task, the group is merely executing the plan.

Specific measures for this type of task:

- Cost and efficiency of performance - speed and errors
- Evidence of performance level changes over time (quality of task outcome but can only be measured for repetitions of task)
- Member satisfaction with the task, the group, their own role

Known research:

- Increased interpersonal interaction does not always lead to higher productivity
- Groups influence their members toward conformity with the group's standards - this may increase or decrease productivity

Example:

A survival task or rescue task in which a group has to perform to achieve a goal: getting back to base or saving individuals.

Suggested capabilities:

- Shared workspace
- N way communication

Task Type 9: (non-McGrath)

Dissemination of information tasks

The task is to distribute information among members of the group. Information can be distributed by group members sharing information with each other or a superior sharing information with others in the group. There may or may not be a question and answer session.

Specific measures for this type of task:

- Shared understanding of information
- Time for distribution
- Audience reached

Examples:

A corporate officer gives a briefing to a division about the new sales strategy.

A professor gives a lecture to a class.

Suggested capabilities:

- 1 way communication
- Feedback channel
- Object displaying
- Summarization capabilities

3.3.1.2 Transition Tasks

Transition tasks are tasks used to move between work tasks. This may include summarizing the outcome of the last task, assigning action items to members of the group, and noting dates for expected completion of assignments. The beginning and ending of any group collaboration involve transition tasks such as taking role, requesting changes to an agenda, locating missing meeting participants. Transition tasks also apply to asynchronous collaborations. A group member may suggest that the e-mail discussions about a particular subject be ended and volunteer to summarize the discussion and distribute this to group members.

We have also defined special startup and shutdown transition tasks to account for the activities associated with getting all members connected to the collaborative session. If the collaborative session is asynchronous, then startup transition tasks will occur during the entire duration of the collaboration as individuals access the collaborative space. A shut down transition task describes the activities that an individual needs to do to exit from the collaborative space.

The beginning of a synchronous collaborative session may include: role taking, reading minutes from a previous session or reviewing the previous session, viewing the agenda and making sure everyone has the necessary documents for a session to begin. Ending a collaborative session may include planning for the next session, saving collaborative documents as needed and shutting down collaborative systems.

A transition task may occur formally or informally, depending on the social protocol that the group is using. Transitions to the next work task occur formally if the chair of the group either moves the group to the next agenda item or the group votes to move to the next item. Informal transitions to the next work task occur when the group moves the discussion rather naturally to another topic or starts another group activity.

Measures of transition tasks:

- Time
- Questions about what just happened
- Questions about what is going to be done next

3.3.1.3 Social Protocols

Social protocols define the ways in which collaborative sessions are conducted. Collaborative sessions may vary from informal sessions to very formal sessions with a chair, an agenda that is strictly followed, and rules of order. Social protocol capabilities support communication between group members; awareness of group members, group activities, group objects; and basic meeting conduct. Table 3 lists some parameters that social protocol requirements take into consideration.

Example measures of social protocol:

- Conflicts in turn-taking
- Awareness breakdowns (questions about):

3.3.1.4 Group Characteristics

Groups have different requirements depending on the makeup of the group, the location of the group members and the time requirements for collaborative sessions. These dimensions are detailed in table 3. In addition to considering the current dimensions of the group, system requirements should also take into account anticipated changes in these dimensions. For example, all members of a task force might start by being co-located but with the knowledge that, in two weeks, half of the group will be remotely located.

3.3.2 Capability Level

The capability level defines basic capabilities that support different types of collaborative tasks. These "collaborative capabilities" provide a means of matching up tasks with services. Each service provides certain collaborative

capabilities. Matching the tasks and services must be done taking into account how well the service supports the capabilities and whether the service is acceptable given the group characteristics.

The capability level can be divided into those capabilities that support work tasks, transition tasks, group characteristics and social protocols.

3.3.2.1 Capabilities for Work Tasks

- Shared workspace
  - Full access to all objects
  - Restricted access
  - Anonymous contributions

Work tasks require objects to work on. CSCW systems can support the following object types:

- Support for object types
  - Static image
  - Sound
  - Temporal image
  - Computational image

For the object types supported, the following capabilities may or may not be supported:

- Visualization of objects
  - Text
  - 2D
  - 3D
- Object manipulation
  - Creating
  - Editing/ modifying

  - Displaying/ viewing
  - Arranging/ organizing
  - Querying/ retrieving
  - Selecting/ filtering
  - Storing
  - Transforming
  - Sending/ distributing
  - Receiving
  - Validating/ proofing
  - Analyzing/ solving
  - Resuming/ undoing
- Object management
  - Coordinating/ controlling
  - Linking
  - Removing
  - Protecting/ locking
  - Managing/ setting
  - Monitoring

3.3.2.2 Capabilities for Transition Tasks

- Collaboration coordination capabilities
  - Summarization capabilities
  - Playback facility
  - Integration
  - Distribution of objects
  - Translation of objects between modalities
- Collaboration planning capabilities
  - Agenda support

- Calendar support
- Meeting notification
- Voting

- Locator capabilities
  - Locate possible collaborators
  - Locate group members
  - Locate objects

3.3.2.3 Capabilities for Social Protocols

- Awareness indicators
  - Abstraction, real → abstract
  - Temporal, past → future
  - Aggregation, snapshot→ summary
  - Locus of control, perceiver → receiver
  - Traceability, individuality→ groupness
  - Information provision, explicit → implicit
  - Perspective, presenter → viewer
- Meeting conduct
  - Support for multiple collaborations including access to and movement between
  - Floor control including chair recognition, turn-taking, changing levels of
  - Document/object access control
  - Synchronize feature
- Communication
  - Side chats
  - Message passing
  - Message leaving

- N way communication
- 1 way communication
- Gesturing, pointing, agreeing, disagreeing
- Feedback channel
- Private communication
- Secure communication
- Private workspace
- Anonymous communication

At the capability level, measures of the various times spent using objects and object manipulations, as well as the various capabilities for transition tasks and social protocols can be taken. Collisions in turn taking and questions about awareness can also be measured. Errors, wrong paths, etc. can be used as indicators of how well the capabilities work.

3.3.3 Service Level

The service level provides a list of the different types of services that can be used in developing CSCW systems. In the future, this level will be expanded to include pointers to threshold values a given service should meet to provide adequate support for various capabilities. A list of basic features for the various services will also be included at this level.

- E-mail
- Chat facility
- Internet connections
- Telephone conversation
- Multicast Audio
- Multicast Video
- Half duplex audio
- Full duplex audio

- White Board
- Shared workspace
- Shared Applications
- Encryption
- Recording
- History mechanism
- Lists of objects, participants, possible collaborators
- Version control
- Simultaneous sessions
- Collaborative space management
- Collaborative space navigation
- Object repository
- Object control
- Import/ export facilities

Service level evaluations are of two types: comparisons to thresholds and feature sets and comparisons between services (of the same type or of different types). Service level measures such as quality of imported image, (readability, discernability, contrast), audio quality (clarity, echo), video quality (contrast, color tone) are also taken and used to compare services.

3.3.4 Technology Level

All the various implementations of the services exist at the technology level along with a description of their performance. For example, a description of an image resolution application would include the transmission time under various network loads, the expected image quality, and the compression factor. Technology evaluations of performance and quality can be outside of a group work context. These measures can be compared to the threshold

measures at the service level to ensure that this implementation meets or exceeds those values.

User interface components and the elements needed to integrate technology building blocks into a unified CSCW system also exist at this level. Usability evaluations can be done at the technology level. However, these evaluations will need to be conducted within group work tasks. That is, users can be given tasks to perform using a specific implementation of a service (i.e., send an e-mail message to another collaborator). The times, keystrokes, errors made by users in performing this task are factored into the usability of the system.

The system developer must supply the technology level descriptions. A template/ checklist can be provided at this level to facilitate this on a service by service basis.

3.4 Summary of Collaborative Framework

The collaborative framework is intended to benefit both researchers and developers of collaborative systems as well as purchasers and users of collaborative systems. The framework is useful in identifying scenarios for a group to use in evaluating a system or in comparing several systems.

The framework can also serve as a collective memory for the CSCW community. Therefore, over time, it is expected that guidelines will appear at the various levels of the framework. These guidelines will suggest capabilities for work tasks and transition tasks, services best suited to certain capabilities, and the appropriate usability, performance, and features levels needed for specific services.

Level	Top-Down	Bottom-Up
Requirement	Identify tasks, social protocols and characteristics Select systems that support these	Given the capabilities that can be supported, select the system which supports the most tasks of the group, provides the best task outcome and best supports the needed social protocols and group characteristics
Capability	For systems using different capabilites, use scenarios to execute tasks using those capabilities and compare. For a single system, determine if the results are acceptable.	Given the services available, select only those systems having capabilities which can be supported using those services
Service	For systems with desired capabilities, compare services if different ones are used. For a single system, determine if the results are acceptable.	Select only those systems that can be supported with the services available
Technology	For systems with the desired capabilities and services, select those with the desired performance and usability thresholds. For a single system determine if the threshold levels are met.	Select only those systems that meet or exceed a desired threshold

Table 1: Evaluation Questions

Meeting conduct	
Chair	Strict, loose, none
Agenda	Strict, modifiable, none
Rules of order	Yes, no
Titles	Yes, names only, anonymous
Floor control	On agenda, yes, [possible only if chair], informal turn-taking, or free-for-all
Hierarchy support	Voting, contributing-restricted, contributing - free access, observing only
Communication needs	
communication	Private→public, 1 way→ n way
Dependency (interaction)	Loose → tight
Level of security needed	None → secret
Awareness support	
Presence	Who ?
Location	Where ?
Activity level	How active ?
Actions	What ?
Intentions	What next ?
Changes	Who did what, where?
Objects	What objects?
Extents	What, how far?
Abilities	What can be done?
Influence	What can be done?
Expectations	What am I to do next?

Table 2: Parameters for Social Protocol

Dimension	**Values**
Time	
Spontaneity of collaborations	planned, spontaneous
Duration of sessions	# hours → days
Time frame for collaborative sessions	synchronous, asynchronous
Location	same for all→ different for all
Group type	
Number of participants	#
Stage of development	newly formed→ working group
Homogeneity	Gender diversity, peer diversity, computer experience diversity, cultural diversity
Length of time group will exist	Limited→ long term
Computer requirements	
Hardware, software requirements	Platforms, resources available (time, money)
Training	Walk-up and use → formal classes
Computer expertise	Novice→ expert

Table 3: Group Characteristics

4 Scenarios for Evaluation of Collaboration Tools

4.1 Introduction

Having defined a framework for collaborative systems and generic task types, we can re-examine what a scenario is, and how we can use it for evaluating collaborative systems.

A *scenario* is an instantiation of a generic task type, or a series of generic tasks linked by transitions. It specifies the characteristics of the group that should carry it out, and the social protocols which should be in place. It describes what the users should (try to) do at the requirements level, but not how they should do it at any of the lower levels of the framework.

Scripts dictate how the users will carry out their tasks, at any of the three lower levels of the framework – the capability, service or technology levels.

Scenarios and the three levels of scripts each serve different, but valuable purposes.

A scenario allows the most natural interaction, of course, since it is not scripted. It can also allow the experimenters to determine whether or not the best ways of doing things are apparent/intuitive to the users.

A script allows a given interaction to be repeated much more exactly. Thus it allows much more accurate comparisons across systems or even across multiple implementations of the same system.

The capability-level script is meant to provide a script that can be used on any system which supports a given set of capabilities.

A service-level script is specific to a certain set of services, but can be used with any implementations of those services. It might be useful in cases where one wants to compare different implementations of the same basic services.

A technology-level script is the most detailed. It tells the users what buttons to press and exactly how to carry out the scenario on a given set of technologies. It provides a detailed, standardized, repeatable sequence of actions. This script can be carried out by even the most inexperienced users, including those lacking the domain knowledge required to run the unscripted scenario as well as those lacking training on the technology being assessed.

It could also serve as training material for new users of the technology. It is conceivable that such a script could even be carried out by participatons, removing the reproducibility problems the human in the loop necessarily introduces. It can be used for some types of user interface analysis, such as GOMS modelling, but of course it cannot be used to assess issues such as intuitiveness, as the users are guided through every step of the interaction.

Since it can be useful to have both scenarios and various levels of scripts for the same collaborative activity, it is instructive to consider how one might generate scripts at various levels of detail for a particular collaborative activity starting from a scenario for that activity. It would be possible to start with the script and have real people (with appropriate expertise, if necessary) run through it on a particular set of technologies, logging the interactions. One could then take the logs, edit out any undesirable actions (such as wrong paths or puzzling over the interface) and create a detailed technology-level script from it. By generalizing the script first to generic services and then to the capabilities supported by those services, the scenario developer could then generate higher-level scripts to allow scripted evaluation across a greater variety of collaborative systems.

4.2 The Beginning of a Scenario Repository

The Evaluation Working Group (EWG) has created some sample scenarios and scripts which may be used for evaluation. They may be found at `http://www.antd.nist.gov/~icv-ewg/pages/scenarios.html`. We encourage others to contribute scenarios or scripts that they have written as well. A scenario template to complete may be found at `http://zing.ncsl.nist.gov/~cugini/icv/domain-template.html`.

Each example clearly sets out the types of tasks, transitions, social protocols and group transitions that it exercises (or aims to exercise). It also lists the capabilities and/or services required or recommended to complete the scenario. This information will be helpful for the user in choosing appropriate scripts and/or scenarios to use for their evaluation, as discussed below.

4.3 Using Scenarios for Evaluation

4.3.1 Using Scenarios to Do Iterative Evaluations of a Single System

One major goal of the Evaluation Working Group is to support system developers in their need to do frequent evaluations of the system to validate the theoretical improvements offered by new versions.

In order to do this the developer should attempt to choose one or more scenarios or scripts that exercise the types of capabilities and services their system provides and/or involves the types of tasks, transitions and social protocols that they hope to support.

Developers who devise their own scenarios or scripts to exercise other sets of capabilities are encouraged to contribute them to the scenario collection that the EWG has started.

4.3.2 Using Scenarios to Evaluate a System's Appropriateness for your Requirements

For potential users looking to select a collaborative system to suit their needs, the scenarios can serve a different purpose. Instead of choosing scenarios based upon a particular set of capabilities or services, they can instead choose scenarios that highlight the types of requirements (tasks, transitions, social protocols and group characteristics) that they need a system to support.

4.3.3 Using Scenarios to Compare Systems

If multiple systems are to be compared using scenarios, several scenarios should be chosen for the evaluation. A complete comparison of the systems should compare the performance of the systems on the different scenarios. The set of scenarios and, by extension, the requirements on which each system performs well or poorly is an indicator of the systems's strengths and weaknesses. In most comparisons, it will be impossible to say which system is best, but it may be possible to determine which which is best for a

particular set of requirements, or which has the best breadth of requirement support even if it isn't the best for any given set of requirements. The results will have to be interpreted in terms of the goals of the experimenter.

5 Measures and Metrics

In the preceding sections, we presented a framework for classifying collaborative systems and an approach to evaluation that employs scenarios. Here we enumerate the measures and metrics introduced in Section 3 and discuss methods for gathering the metrics.

The measures and metrics described here are relevant to laboratory experiments. The goal of the Evaluation Working Group is to define inexpensive evaluation technology that can be reproduced in a laboratory. Hence, the measures will not address organizational impact and other measures that require fielding production-quality systems.

Experiments involving human subjects, or sets of subjects, are expensive and time consuming, both in terms of collection and analysis. In many cases, the measures must be developed and validated before they can be applied with confidence. Despite these difficulties, we lay out here some options for evaluating collaborative systems at all four levels of the framework discussed in section 3.

We begin with a graphical overview of the methods, metrics and measures, and then we present the measures found at each level of the framework. Metrics will be introduced in the context of measures and then described further in their own section. Finally, we will describe the methods used to collect the metrics we have identified.

5.1 Overview

The diagram below illustrates the relationship of data collection methods, metrics, measures, and human judgment. The diagram also shows how the measures map onto the four levels of the collaborative framework.

At the bottom, we use data collection methods, such as a logging tool or video taping, to gather metrics on a system. Metrics are indicators of system, user, and group performance that can be observed, singly or collectively, while executing scenarios. Metrics - like time, length of turn, and other countable events - are directly measurable and can be collected automatically.

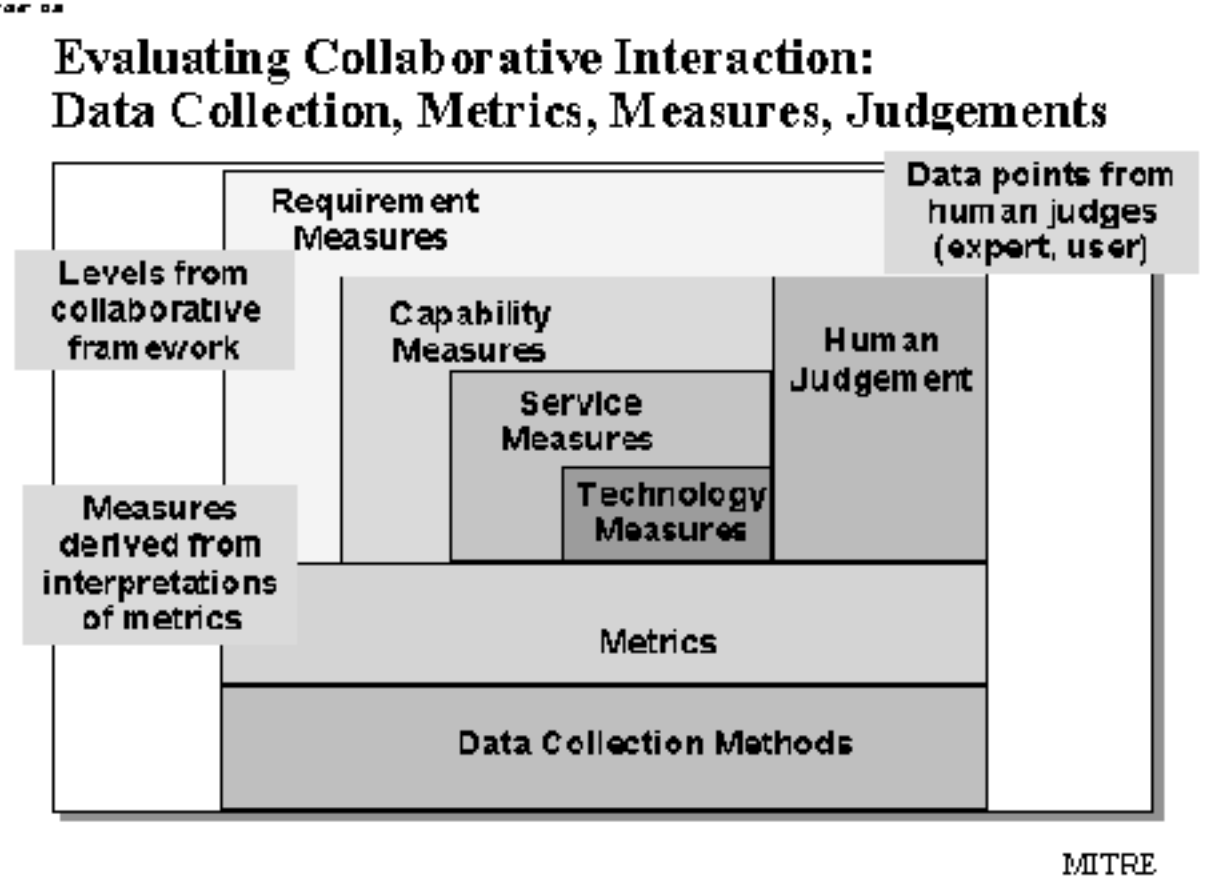

Figure 2: Overview of the levels of measures

Measures can be taken at each of the four levels of the collaborative framework. Measures are derived from interpretations of one or more metrics. For example, task completion time (a requirement-level measure) is based on start and end time metrics. A measure can also be a combination of interpreted metrics and other measures. A complicated measure, like efficiency, is partially derived from the interpretation of metrics like time, user ratings, and tool usage. In addition, measures of system breakdown (taken at the service level) contribute to efficiency.

A simple way to distinguish between metrics and measures is by the following statement: a metric is an observable value, while a measure associates meaning to that value by applying human judgment.

Human judgment is important as humans themselves can be a source of data. Questions to both experts and users of the system can provide valuable data points in the evaluation of a system.

Now we are going to shift perspective and discuss, in greater detail, the measures, metrics, and methods from the top down.

5.2 Measures

The framework described in section 3 of this document divides the collaborative system design space into four levels: requirement, capability, service, and technology. Evaluations may be conducted at all of these levels. By this we mean that we have identified measures that are associated with each level. For example, participation is a requirement level measure correlated with various metrics: the total number of turns per participant, the total number of turns per group, and user ratings. By contrast, usability is a technology level measure.

For each measure, we present:

- a definition
- the "building blocks": metrics and other measures
- associated task types

The "building blocks" of a measure are metrics and other measures. A build-

ing block can actually be a combination of metrics or arithmetic formulas involving metrics. It is important to note that the authors are hypothesizing on many of the metric and measure components of each measure. Where available, we have included references to supporting research. For some measures, we have also included additional metrics and measures that might show some indirect relevance. Metrics are more fully detailed and discussed in subsection 5.3.

In general, all measures can be applied to all tasks. However, some measures may have little or no relevance for a particular instantiation of a task type. For example, measuring participation may not be important to a dissemination of information task (Type 9), whereas it appears to be very important in a brainstorming and creativity task (Type 2). With each measure, we include a list of what we believe to be the applicable task types that the measure helps to evaluate. The task types are enumerated and explained in section 3 of this document.

When discussing the requirement and capability level measures, we will also address the four aspects of group work: work task, transition, social protocol, and group characteristics (see section 3).

5.2.1 Requirement Level Measures

The requirement level measures include:

R1. task outcome

R2. cost

R3. user satisfaction

R4. scalability

R5. security

R6. interoperability

R7. participation

R8. efficiency

R9. consensus

R1. Task Outcome

Task outcome is a measure of the state of a particular task. There is a set of artifacts, produced during execution of the task, such as documents, ideas, solutions, and defined processes.

Aspects of group work evaluated

- work task

Metric and measure components

See the different task types in section 3 for specific metrics and measures in addition to the general ones list below.

- countables - number of generated artifacts
- task completion (yes/no)
- expert judgments of product quality
- user ratings of product quality

Additional metrics and measures that may be relevant or correlated

- repair activities
- grounding (This is a measure defined at the capability level.)

Associated task types

- all

R2. Cost

Cost is the measure of time invested in the system and the resources consumed in executing an activity.

Aspects of group work evaluated

- work task
- transition

Metric and measure components

- preparation cost measures - monetary amount and learning time
- countables - number of turns
- length of turns
- execution time

Additional metrics and measures that may be relevant or correlated

- repair activities
- breakdown (This is a measure defined at the service level.)

Associated task types

- all

R3. User Satisfaction

User satisfaction is a subjective measure of satisfaction with respect to the four aspects of group work.

Aspects of group work evaluated

- work task
- transition
- social protocol

Metric and measure components

- user ratings

  General questions about:

  . satisfaction with the group process
  . satisfaction with the task outcome or the final solution
  . satisfaction with an individual's participation
  . satisfaction with the group participation

Additional metrics and measures that may be relevant or correlated

- countables - number of generated artifacts
- execution time
- turn overlap (ability to interrupt)
- repair activities
- breakdown (This is a measure defined at the service level.)
- grounding (This is a measure defined at the capability level.)
- awareness (This is a measure defined at the capability level.)

Associated task types

- all

R4. Scalability

Scalability is the measure of a system's accommodation for larger or smaller group size.

Aspects of group work evaluated

- social protocol
- group characteristics

Metric and measure components

- time to complete particular task versus number of users
- resources needed to complete particular task versus number of users
- expert judgments - yes/no, to what degree

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all task types with the exception of mixed motive task (Type 6)- bargaining subtype - which is limited to two parties (possible collaboration within the parties is associated with a separate task)

R5. Security

Security is a measure of the protection of information and resources.

Aspects of group work evaluated

- social protocol
- group characteristics

Metric and measure components

- expert judgments

  General questions about:

  . system security
- yes/no to list of features, to what degree

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

R6. Interoperability

Interoperability is a measure of how well system components work together, sharing functionality and resources.

Aspects of group work evaluated

- work task
- transition
- group characteristics

Metric and measure components

- expert judgments - yes/no to list of features, to what degree
- tool usage - Which tools were used and how often? Was the optimal set of tools used?

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

R7. Participation

Participation is the measure of an individual's involvement in a group activity.

Aspects of group work evaluated

- work task
- transition
- social protocol
- group characteristics

Metric and measure components

- countables

  Tsai (1977) has found that participation measures involving three different definitions of a unit act are highly correlated:

  . number of sentences

  . number of floor turns (regardless of length)

  . a unit based on the category of the act

  An individual's participation, P_i, can be measured as the total of any one of these unit acts, t_i, divided by the total number of unit acts in the group, $t_{1+2+...+n}$, where n is the number of group members. $P_i = t_i / t_{1+2+...+n}$

  Group participation can be calculated by the number of contributing participants divided by the number of total participants.

  Equality of participation is 100% if each group member has the same value of participation: $P_1 = P_2 = ... = P_n = 100/n => 100\%$ equality of participation.

- user ratings

  General questions about:

  . satisfaction with an individual's participation

. satisfaction with the group participation

Additional metrics and measures that may be relevant or correlated

- turn overlap
- conversational constructs
- grounding (This is a measure defined at the capability level.)

Associated task types

- Type 1: planning tasks
- Type 2: brainstorming and group creativity tasks
- Type 3: intellective tasks
- Type 4: decision making tasks
- Type 5: cognitive conflict tasks
- Type 6: mixed motive tasks
- Type 7: competitive tasks

Less-associated task types

- Type 8: non-competitive contests
- Type 9: dissemination of information tasks

R8. Efficiency

Efficiency is a measure of group and system effectiveness and productivity.

Aspects of group work evaluated

- work task

- transition
- group characteristics

Metric and measure components

- percent efficiency = (task time - repair activities time)/execution time
- percent efficiency = countable (number of generated artifacts) per unit of time
- user ratings

  General questions about:

  . efficiency
- tool usage - Was the optimal set of tools used?
- breakdown (This is a measure defined at the service level.)

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

R9. Consensus

Consensus is the measure of general agreement or group unity in outcome.

Aspects of group work evaluated

- work task
- transition

Metric and measure components

- user ratings

  General questions about:

  . agreement with the task outcome

- grounding (This is a measure defined at the capability level.)

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- Type 1: planning tasks
- Type 3: intellective tasks
- Type 4: decision making tasks
- Type 6: mixed motive tasks - negotiation and bargaining subtypes

5.2.2 Capability Level Measures

Capability level measures evaluate general capabilities of collaborative systems (see Section 3). The capability measures include the following:

C1. awareness

C2. collaboration management

C3. human to human communication

C4. grounding

C5. collaborative object support

C6. task focus

C7. transition

C1. Awareness

Awareness is the degree of "having ... realization, perception, or knowledge" (Webster) of surroundings and events. For example, awareness of:

- other participants, their roles, etc.
- actions (pointing, speaking, annotating, etc...)
- objects and object manipulations
- social protocols

Aspects of group work evaluated

- work task
- transitions
- social protocol
- group characteristics

Metric and measure components

- user ratings

  General questions about:

  . awareness of other participants, their roles, etc.
  . awareness of actions
  . awareness of objects and object manipulations

- conversational constructs

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

C2. Collaboration Management Measures

The set of collaboration management measures assesses support for coordinating collaboration.

These measures are used to evaluate the following (non-exhaustive) list of capabilities:

- support for multiple collaborations
- floor control
- agenda support
- document access control
- collaborator access control
- synchronize feature

Aspects of group work evaluated

- work task
- transition
- social protocol

Metric and measure components

- expert judgments - yes/no, to what degree

Additional metrics and measures that may be relevant or correlated

- turn overlap (including interruptions)

Associated task types

- all

C3. Communication (human to human)

Communication is a measure of the exchange of information in any of the following forms:

- verbal (spoken or written)
- visual
- physical

Aspects of group work evaluated

- work task
- transition
- social protocol

Metric and measure components

- countables - number of turns per participant
- turn overlap (both simple overlap and interruptions are important)
- expert judgments

  General questions about:

  . availability of communication
  . goodness of communication
  . ability to get floor control
- yes/no, to what degree
- user ratings

  General questions about:

  . goodness of communication
  . getting floor control

. getting the attention of other participants
. ability to interrupt

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

C4. Grounding

Grounding is a measure of how well common understanding is established.

Aspects of group work evaluated

- work task
- transition
- social protocol

Metric and measure components

- user ratings
  General questions about:
  . reaching common understanding with other participants
- number of turns
- length of turns
- turn overlap
- conversational constructs

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task type

- Type 1: planning tasks
- Type 3: intellective tasks
- Type 4: decision making tasks
- Type 5: cognitive conflict tasks
- Type 6: mixed motive tasks - negotiation and bargaining subtypes

C5. Collaborative Object Support

The set of measures assesses support for collaborative objects.

These measures are used to evaluate the following (non-exhaustive) list of capabilities:

- object manipulation and management
- shared workspace

Aspects of group work evaluated

- work task
- transition

Metric and measure components

- expert judgments - yes/no, to what degree
- tool usage - Was the optimal set of tools used?

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

C6. Task Focus

Task focus measures the ability to concentrate on the task at hand.

Aspects of group work evaluated

- work task
- social protocol

Metric and measure components

- task focus = (overall time - transition time - other time) / overall time

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

C7. Transition Measures

This set of measures assesses support for transitions.

These measures are used to evaluate the following (non-exhaustive) list of capabilities:

- collaboration startup
- summarization
- playback
- archiving
- object exporting and importing
- distribution of objects
- translation between modalities
- meeting notification

Aspects of group work evaluated

- time
- transition

Metric and measure components

- expert judgments - yes/no, to what degree
- user ratings
  General questions about:
  . flow of transitioning between tasks
- conversational constructs

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all but Type 7: competitive performances

5.2.3 Service Level Measures

Service level measures evaluate the services provided by a collaborative system. This type of measure includes:

S1. breakdown

S2. tool usage

S1. Breakdown

Breakdown measures how often the user has to rationalize a problem experienced. Breakdowns can occur in communication, in coordination, in the system components, etc.

Metric and measure components

- conversational constructs
- repair activities
- tool usage - Which tool was used?

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

S2. Tool Usage

Tool usage is the degree to which the optimal tools are used for a particular task.

Metric and measure components

- tool usage - Which tools were used and how often?
- expert judgments - Were the 'right' tools used?

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

5.2.4 Technology Level Measures

Measures on the lowest level of the framework, the technology level, evaluate the implementation of a collaborative system. The technology consists of software or hardware components, interfaces, and component connections. The measures include:

T1. usability

T2. specific technology

T1. Usability

Usability measures evaluate the ease, accessibility, and intuitiveness of the specific graphical user interfaces of the system tools and components. Since usability evaluation is done on specific graphical user interfaces, the usability measures are realized at the technology level although the component measures are also based, in part, on measures taken at different levels of the framework.

Metric and measure components

- expert judgments - standard user interface evaluation questions
- user ratings - standard user interface evaluation questions

- tool usage - Which tools were used and how often? Was the set of optimal tools used?
- repair activities
- breakdown (This is a measure defined at the service level.)
- awareness (This is a measure defined at the capability level.)

Additional metrics and measures that may be relevant or correlated

- to be determined

Associated task types

- all

T2. Specific Technology Measures

(EWG contributors are preparing a list of pointers to industry standards)

5.3 Metrics

The following list of metrics comprises the observable data elements that we have identified as useful indicators of system and group performance. A single metric is just a number (e.g., number of turns) or a set of numbers (e.g., start time and end time).

Metrics can also be used in combinations (e.g., the repair activities metric is partially built up from the number of undos and the number of cancels and is the deviation from some established 'right path').

Metrics are associated with an interpretation in order to build measures. For example, the measure of task completion time is the interpreted end time of a task minus the interpreted start time of the task.

As another example of the metric and measure relationship, let's consider the turn overlap metric. Overlap of speaker A by speaker B can be counted

if the start time of speaker B is greater than the start time of speaker A but less than the end time of speaker A. Further interpretation is required to determine if a particular occurrence of turn overlap is an attempt to gain the floor (interruption in communication) or a back-channel response indicating attentiveness (support, grounding).

For each metric, we present:

- a definition
- guidelines for observing or recording the metric

Where applicable, the metrics are enumerated or they are broken down into finer granularity.

We attempt to provide some guidelines for gathering each metric. It is important for the evaluator to record how each metric was observed so that comparisons can be made with data collected from multiple experiments. Methods used to gather each metric are listed at the end of this section.

Some metrics are:

M1. countables

M2. length of turns

M3. task completion

M4. time

M5. preparation cost metrics

M6. expert judgments

M7. user ratings

M8. tool usage

M9. turn overlap

M10. repair activities

M11. conversational constructs

M1. Countables

Countables are any items that can be counted, once a clear definition has been developed. The difficulty with countables is in ensuring that the definitions are consistent and that items are counted in the same manner.

Examples of countables:

- turns (spoken utterances, e-mails, typed lines, etc.)
- moves (a subtype of turn, e.g., turns taken in a game)
- steps (number of high-level steps to produce result; number of mouse clicks or button presses towards accomplishing a task)
- trials (number of aborted attempts and successful attempts to reach goal)
- ideas generated
- responses (replies to e-mail message; responses to a question)
- cancel button presses (Note that there could be different reasons for this: 1) the user made a mistake, or 2) the user changed his/her mind)
- ...

M2. Length of Turns

Length of turns refers to the length or size of any type of turn. A turn is a single unit of communication or activity.

For example:

- length of spoken utterance = end time (of utterance) - start time
- length of e-mail message = number of lines
  - or - = number of sentences
  - or - = number of words
  - or - = size in bytes

- length of turn in chat session = number of sentences

  - or - = number of words

- length of move (subtype of turn) = time at end of move - time at start of move

M3. Task Completion

Task completion refers to whether or not a task is completed. In some cases, the degree of completion can be measured.

M4. Time

Time, as a metric, supports a number of measures. There are base metrics for start and end time which can be interpreted to support the list of measures that follows. Time can be measured with respect to an individual, as a sum of all individual times, and/or as the longest individual time in the group.

- Overall task execution time

  Overall task execution time is the interval from task beginning to task end. This is the total time it takes the collective group to complete execution of the task.

- Task time

  Task time is the time spent on the actual task. This does not include transition time or time spent doing non-task-related activities.

- Transition time

  Transition time is the amount of time it takes the group to transition from one task to another. This includes set up time.

- Other time

  Other time is determined by the following formula:

  other time = overall time - (task time + transition time)

- Repair activities time

  Repair activities time is the time spent going down a wrong path in addition to the time spent repairing those actions. (See the definition of repair activities below.)

  In asynchronous tasks, repair activities time = overall time - task time - transition time - idle time

Time may also be broken down into finer granularity, if desired. For example, time spent on each type of object manipulation can be used as a metric.

M5. Preparation Cost Metrics

Preparation cost metrics include the monetary amount of a system and the learning time for individual users.

M6. Expert Judgments

Expert judgments are questions posed to experts.

General questions about:

. Yes/no/to what degree/quality questions relating to:

  - scalability
  - security
  - interoperability
  - collaboration management measures
  - communication
  - collaborative object support measures
  - transition measures

. usability

. product quality

. task outcome

. the set of tools used to accomplish the task (the efficiency of the tools used)

M7. User Ratings

User ratings are questions posed to users.

These include general questions about:

- Task outcome
  - . product quality
- User satisfaction
  - . satisfaction with the group process
  - . satisfaction with task outcome or final solution
  - . satisfaction with an individual's participation
  - . satisfaction with the group participation
- Participation
  - . an individual's participation
  - . group participation
- Efficiency
  - . efficiency of the system
- Consensus
  - . consensus on the solution
  - . consensus on the task outcome
- Awareness
  - . awareness of other participants
  - . awareness of objects
  - . awareness of actions
- Communication

    - whether communication was possible
    - the goodness of the communication
    - ability to get floor control
    - ability to ask a question / make a response
- Grounding
    - establishing common understanding with other participants
    - understanding what other participants were talking about
- Transition
    - smoothness of the transitions
- Usability
    - standard user interface evaluation and usability questions

M8. Tool Usage

Tool usage is the frequency and distribution of tools used to accomplish a particular task.

Tool usage is also the way in which tools are used to accomplish a particular task. This can be measured as the deviation from a pre-determined correct tool usage established by experts. Experts can rate the use of sets of tools for each subtask, and tool usage can be measured as the deviation from those ratings.

M9. Turn Overlap

Turn overlap is the overlap in communication by two or more people.

Turn overlap occurs when the start time of a turn happens before the end time of a previous turn. Turn overlaps can be counted.

M10. Repair Activities

Repair activities include all errors, following down a wrong path, and all actions performed to correct those errors. Repair activities also include not knowing what to do and seeking help. In order to have this metric, there must be an established 'right path', or a list of possible 'right paths'. Experts determine the 'right paths'. The repair activities metric is made up of the number of undos and the number of cancels, as well as repetitive or useless keystrokes.

M11. Conversational Constructs

Conversational constructs is a general category of metrics that includes semantic and grammatical content of communication.

Topic segmentation and labeling is the ability to segment dialog into multiple, distinct topics with each segment labeled as to its topic.

Given topic segmentation, we can measure:

- topic mention - the number of times a topic is mentioned
- distance of topic from a given turn

A reference is the use of a grammatical construction (e.g., a definite noun phrase or a pronoun) to refer to a preceding word or group of words. References can be counted by their occurrence, type, and distribution in a transcript. The use of pronouns can indicate group inclusion (you, we) or exclusion (I).

5.4 Data Collection Methods

The following is a list of methods, or data-collection instruments, used for gathering the metrics described above:

- logs
- direct observation

- questionnaires/interviews/rating scales (open-ended or closed/fixed alternatives)
- tape recorder / transcription
- video recorder / annotation

Many of the metrics can be gathered by a variety of tools or methods. Each method offers potential opportunities not available by other means but also has inherent limitations. For this reason, we suggest using multiple methods to obtain each metric. The possibilities are enumerated in the matrix below.

metrics	*logs*	*observations*	*questionnaires*	*audio*	*video*
countables	x	x		x	x
expert judgments		x	x		x
length of turns	x	x		x	x
turn overlap	x			x	
resource costs	x	x	x	x	x
task completion	x	x			
time	x	x			x
tool usage	x	x	x		x
user ratings			x		
conversational constructs	x	x		x	
repair activities	x	x	x		

For example, countables can be measured by studying logs, observations, audio transcripts, and video recordings.

5.4.1 Methodology: Logging

Motivation

In our evaluation of many of the above items, the need frequently arises for collection of data about running systems. Because videotaping every session can be cumbersome and costly, the ability to log automatically many aspects of the interaction will be important in developing easy, repeatable, evaluation suites. To this end we require a multi-modal logging capability to support our evaluation efforts.

A second motivation for logging is that logged data can be used for feedback and training of adaptive systems; however, this aspect will not be explored further here.

A limitation of the logging approach to data collection is that we can only log what happens on the workstation, or other actions the workstation is "aware" of.

The MITRE Multi-Modal Logger

To support our evaluation efforts, MITRE is providing a tool which can be used to log multi-modal data while collaborative systems are running. This data can be fine-grained (individual X events, for instance) or coarse-grained (a record of which windows the user interacts with); automatically gathered (via instrumentation) or manually created (via post-hoc annotation); and at the level of the physical or virtual device (say, X events or the output of /dev/audio), the interface (say, a record of menu selections made and the content of text entry fields), or the application (a record of actions taken, such as a retrieval of information or a command issued). This information can also be in a variety of modalities, such as text, images, and audio.

Since this information will typically be gathered for multiple users and multiple interactions with the system in question, the notion of a "trial" or "session" is supported. In addition, each trial might require information to be gathered from multiple components simultaneously (say, when a speech recognizer is used in conjunction with an independent multi-modal system). Therefore, sharing each trial among multiple components, potentially running on different hosts, is also supported.

MITRE's multi-modal logger provides an easy-to-use API for instrumenting existing applications; it embodies a database structure which groups data points by application, and applications by session; it supports the typing of data points via MIME types; and it provides a set of tools for reviewing and annotating data collected via instrumentations.

Method, Granularity and Level of Data Collection: MITRE's logger offers a simple solution to the question of what granularities and levels of data may be collected: the instrumenter inserts whatever logger calls are

desired into the source code and is thus in complete control of where, in the code, the log entries are generated, how many are generated, and what data types are assigned to them. Anything that one has access to in the source code can be logged.

It is undeniable, of course, that in most cases, access to the source code is required. However, this is not always true. For instance, if we want to log windowing events, tools such as X-Runner, which "instruments" X applications by substituting its own version of the Xt library, will soon generate windowing events for consumption by other applications (such as our logger). In this case, we only need to link against the new Xt library, rather than needing access to the source code for a full compilation.

Availability: The logger is available in a number of programming languages. While the API is written natively in C (and thus also available in C++), MITRE has already created bindings for Python and for the LambdaMOO programming language used in our Collaborative Virtual Workspace; bindings for Tcl are also planned.

The logger supports both mSQL and "flat" databases. mSQL is a lightweight RDB implementation which is free for many types of non-commercial use. If this is not available, the "flat" option can be used, which causes the logger system to create and maintain its own database in a group of flat binary files.

Log Records and Data Types: Log records are grouped by a session name, and within a session, by application name. A number of different applications may log to the same session. Time stamps are automatically included in the log records.

Each log record sent to the database must contain a data type, which may be associated with a MIME type. The data type describes the data, and the MIME type allows other applications to use it in intelligent ways, such as playing audio or video files correctly.

The logger system keeps a table of predefined data types, to which users may add (or delete) types. A data type includes a name, an optional associated MIME type, and a flag indicating whether the actual data or a pointer to a file containing it will be stored in the database.

Log Review and Annotation: MITRE is also developing a log review and annotation tool to distribute with the logger. It allows users to view the data that has been logged for a given session. One may view all the data or select a subset by application, datatype and/or timestamp. The data is displayed along a scrollable timeline, with the data sorted into streams by application and datatype.

This tool may also be used to add post-hoc annotations to the data, linking sets of points to user-produced text. The annotations will also appear as datapoints in the log session, and may themselves have further annotations applied to them.

There is also a replay facility which will allow the reviewer to replay the logged interaction in approximately real time. Currently an initial implementation exists, which will be improved and revised as users begin to provide feedback.

6 Evaluating Interaction: Where to Go?

Today, many research communities face the problem of how to evaluate interaction. To name a few:

- Spoken language systems can only evaluate "pre-recorded" speech input vs. database tuples as a response; this methodology is effective because all systems receive the identical input, factoring out subject variability – at the expense of interaction: interaction is disallowed because there is no user in the loop – you cannot talk back to a pre-recorded message.
- Interactive information retrieval search has also, to date, failed to define an evaluation paradigm because we do not really understand how to compare user A interacting with system X with user B on system Y.
- Interface evaluation is limited to high-level qualitative results, from, e.g., a cognitive walk-through, or to expensive laboratory and field evaluations involving significant numbers of subjects.
- Collaborative environments offer an even bigger challenge: evaluating interaction among *multiple* participants, who may differ in rank, preferred mode of usage, familiarity with the system, etc.

In all of these cases, the difficulties are similar: a multi-dimensional interface space, the need to factor in (or out) user variability, and lack of definition of the task space (or task accomplishment metrics).

6.1 A Possible Solution Path

For all of these tasks, we can abstract away from the specifics of real users, actual GUIs and specific tasks to a level of interaction primitives: a basic set of activities that occur during most kinds of interaction. These might include:

- Signaling readiness to participate (joining a session),
- Signaling end of participation (leaving a session),

- Communicating to participants,
- Responding to communication from participant(s),
- Signaling successful communication (acknowledgement),
- Signaling failure to communicate (error condition), and
- Attempt to interrupt.

The next step is to build some models of prototypical collaborative tasks using such primitives. For example, the characterization of the use of a collaborative environment to schedule a meeting:

- Participant sends either schedule or suggested time
- Participant acknowledges agreement or conflict or suggests alternative(s)
- This process iterates until someone identifies a consensus or a plurality This activity is made up of iterated communications, acknowledgements, responses, etc. This particular activity is normally asynchronous, so that interruptions don't really enter in. However, a successful collaborative system must support these asynchronous communications. Obviously, in this case, e-mail would work fine (and may be the baseline system to beat). However, the way in which a user's on-line schedule can be shipped around and be made viewable, may decrease the amount of time any one user has to take, how many iterations are needed, and/or improve the quality of the solution.

Let's take another example: a briefing in a collaborative environment. This would require:

- Participants entering a session at a fixed time,
- Briefer able to broadcast briefing materials, well synchronized in case of multiple media (e.g., talk and point to viewgraph),
- Participants able to join late, leave early,
- Participants able to interrupt and ask questions,

- Briefer able to answer questions,
- Participant able to enter after the briefing and replay it.

Again, these activities can be broken down into participation, broadcast, response, acknowledgement, etc, but now in a generally synchronous environment. A good environment would allow these things to happen with minimal disruption to the briefing, maximal bandwidth for the briefing, ability to support and control interaction (what happens if two people ask questions at the same time?), etc.

Suppose we can collect data to characterize these tasks at this level of abstraction. Can we then write a basic "collaborative briefing script" that simulates a collaborative briefing at this level of abstract event, such that it can be "enacted" on a variety of systems? This would have the advantage of 1) factoring out human variability – each system would be benchmarked against the same script; 2) providing "real-time" evaluation – the interaction would be on a session basis, so would not take longer that the actual session (an hour for the briefing – for meeting scheduling, real time could be speeded up, as long as asynchronous communication is demonstrated). And the same measurements would be made for each scripted event, and the same outcome metrics reported, e.g., system A could not support "brief and point", system B could not support interruptions, system C allowed interruptions, but the questioner "drowned out" the speaker, so part of the briefing was lost, system D had so much delay (5 seconds) between sending a question and receipt of question that the context was lost, etc.

The scripts may be initially enacted by a set of live testers, each following their part in the script, testing multiple systems. Longer term, it might be possible (though not necessarily advisable) to build "participatons" or automated agents that could "push the buttons" required for system testing. Each system would require its own participaton, but the development of such testing scripts and agents would enable system developers to iteratively test their own systems during development.

Even if we cannot build fully automatic agents to simulate human participation, asking these questions pushes us to examine the sources of variability in these tasks. The scenario-based method that we have outlined in this document provides a framework for experimentation in this direction, even while we explore the more conventional means of scenario-based evaluation.

7 Bibliography

Bass, L. (1996) "Mini-workshop: scenarios for CSCW systems." In Engineering for Human-Computer Interaction: proceedings of the IFIP TC2/WG2.7 working conference on engineering for human-computer interaction, Yellowstone Park, USA, August 1995, editors Bass, L. J. and Unger, C. Champman & Hall, 1996, pp. 333-338.

Grudin, J. (1988), Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces, in Proceedings of ACM CSCW'88 Conference on Computer-Supported Cooperative Work, Perspectives on Evaluation, pp. 85-93.

McGrath, J. E. (1984), "Groups: Interaction and Performance," Englewood Cliffs, N. J., Prentice-Hall.

Pinsonneault, A. and Kraemer, K. (1993), "The Impact of Technological Support on Groups: An Assessment of the Empirical Research" in Readings in Groupware and Computer Supported Cooperatvie Work, edited by Baecker, pp. 754-773.

Nardi, B. A. (1995) "Some reflections on scenarios." In Scenario Based Design: Envisioning Work and Technology in System Development, edited by Carroll, J. M. pp. 397-399.

Potts, C. (1995) "Using schematic scenarios to understand user needs," in DIS95, Ann Arbor, MI, pp. 247-256.

Tsai, Y. (1977) "Hierarchical structure of participation in natural groups," Behavioral Science, vol. 22, pp. 38-40.

Watts, L., Monk, A., and Daly-Jones, O. (1996) "Inter-Personal Awareness and Synchronization: Assessing the Value of Communication Technologies." International Journal of Human-Computer Studies, vol. 44, no. 6, pp. 849-873.

Webster's Ninth New Collegiate Dictionary. Merriam-Webster Inc., 1985.

A ICV Team

NAME	ORG	PHONE	E-MAIL
Samuel Bayer	MITRE	617.271.2035	sam@linus.mitre.org
Lloyd Brodsky	AMERIND	703.836.5900	lloyd_brodsky@amerind.com
Mark Carson	NIST	301.975.3694	mark.carson@nist.gov
John Cugini	NIST	301.975.3248	cuz@nist.gov
Laurie Damianos	MITRE	617.271.7938	laurie@linus.mitre.org
Jean-Philippe Favreau	NIST	301.975.3634	jfavreau@nist.gov
Andy Greenberg	NIMA	202.863.3121	gvhc92a@prodigy.com
Lynette Hirschman	MITRE	617.271.7789	lynette@mitre.org
Anne R. Humphreys	VWR	412.268.1701	ah34+@andrew.cmu.edu
Craig W. Hunt	NIST	301.975.3827	craig.hunt@nist.gov
Robyn Kozierok	MITRE	617.271.2870	robyn@mitre.org
Jeff Kurtz	MITRE	617.271.2291	jkurtz@mitre.org
Ron Larson	DARPA	xxx.xxx.xxxx	rlarson@darpa.mil
Sharon Laskowski	NIST	301.975.4535	sharon.laskowski@nist.gov
Kevin Mills	DARPA	301.975.3618	klmills@darpa.mil
Doug Montgomery	NIST	301.975.3630	dougm@nist.gov
Jim Morris	CMU	412.889.0419	james.morris@cmu.edu
Jane Mosier	MITRE	617.271.7451	jmosier@mitre.org
Chris Neuwirth	CMU	412.268.8702	cmn+@andrew.cmu.edu
Beatrice T. Oshika	MITRE	703.883.3357	bea@mwunix.mitre.org
Mudumbai Ranganathan	NIST	301.975.3664	mudumbai.ranganathan@nist.gov
Susan Harkness Regli	CMU	412.268.7177	shr1@cmu.edu
David Sanders	DynCorp	703.284.8202	dsanders@snap.org
Jean Scholtz	NIST	301.975.2520	scholtz@zing.ncsl.nist.gov
Charles Sheppard	NIST	301.975.3269	csheppard@nist.gov
Rachid Sijelmassi	NIST	301.975.5064	sijel@snad.ncsl.nist.gov
Brett Strausser	NIST	301.975.3620	brett.strausser@nist.gov
Kim Walls	NIMA	202.863.3121	kimtw@ucia.gov

Table 4: Team Members, Phone and E-mail